RATINGS PREDICTION

Submitted by:

Praveen Pandey

ACKNOWLEDGMENT

I would like to express my gratitude to my guide Shwetank Mishra (SME, Flip Robo) for his constant guidance, continuous encouragement and unconditional help towards the development of the project. It was he who helped me whenever I got stuck somewhere in between. The project would have not been completed without his support and confidence he showed towards me.

Lastly, I would l like to thank all those who helped me directly or indirectly toward the successful completion of the project.

INTRODUCTION

- ## Business Problem Framing

Internet is the best source nowadays for any organisation to know public opinions about their products and services. Many consumers form an opinion about a product just by reading a few reviews. Online product reviews provided by consumers who previously purchased products have become a major information source for consumers and marketers regarding product quality. Research has shown that consumer online product ratings reflect both the customers' experience with the product and the influence of others' ratings. Websites prominently display consumers' product ratings, which influence consumers' buying decisions and willingness to pay.

The opinion information is very useful for users and customers alike, many of whom typically read product or service reviews before buying them. Businesses can also use the opinion information to design better strategies for production and marketing. Hence, in recent years, sentiment analysis and opinion mining have become a popular topic for machine learning and data mining.

We have a client who has a website where people write different reviews for technical products. Now they are adding a new feature to their website i.e. The reviewer will have to add stars(rating) as well with the review. The rating is out 5 stars and it only has 5 options available 1 star, 2 stars, 3 stars, 4 stars, 5 stars. Now they want to predict ratings for the reviews which were written in the past and they don’t have a rating. So, we have to build an application which can predict the rating by seeing the review.

- ## Conceptual Background of the Domain Problem

A recent survey (Hinckley, 2015) revealed that 67.7% of consumers are effectively influenced by online reviews when making their purchase decisions. More precisely, 54.7% recognized that these reviews were either fairly, very or absolutely important in their purchase decision

making. Relying on online reviews has thus become a second nature for consumers.

Consumers want to find useful information as quickly as possible. However, searching and comparing text reviews can be frustrating for users as they feel submerged with information. Indeed, the massive amount of text reviews as well as its unstructured text format prevent the user from choosing a product with ease. The star-rating, i.e., stars from 1 to 5 on online platform, rather than its text content gives a quick overview of the product quality. This numerical information is the number one factor used in an early phase by consumers to compare products before making their purchase decision.

Generally, the ratings and the price of the product are simple heuristics used by the customers to decide over the final purchase of the product. But often, the overall star ratings of the product reviews may not capture the exact polarity of the sentiments. This makes rating prediction a hard problem as customers may assign different ratings for a particular review. For example,

For instance, a user may rate a product as good and assign a 5- star score while another user may write the same comment and give only 3 stars. In addition, reviews may contain anecdotal information, which do not provide any helpful information and complicates the predictive task.

The question that arises is how to successfully predict a user's numerical rating from its review text content. One solution is to rely on supervised machine learning techniques such as text classification which allows to automatically classify a document into a fixed set of classes after being trained over past annotated data.

Different users may have different sentiment expression preferences. For example, some users choose to use "good" to describe a just-so-so product, but others may use "good" to describe an excellent product. Beside the user bias, there is also a product bias. We may use different opinion words to review different products, or even use the same opinion word to express different sentiment polarities for different products; for example, the opinion word "long" can be express a "positive" feeling for cell phone's battery life, but may have a "negative" feeling for a camera's focus time. Therefore, it is important to consider the relationship

between the review-authors, as well as that of the target products, for review rating prediction.

- ## Review of Literature

  According to the Lackermair, Kailer and Kenmas, product reviews and ratings represent an important source of information for consumers and are helpful tools in order to support their buying decisions. They also found out that consumers are willing to compare both positive and negative reviews when searching for a specific product. The authors argue that customers need compact and concise information about the products. Therefore, consumers first need to pre-select the potential products matching their requirements. With this aim in mind, consumers use the star ratings as an indicator for selecting products. Later, when a limited number of potentials products have been chosen, reading the associated text review will reveal more details about the products and therefore help consumers making a final decision.

  It becomes daunting and time-consuming to compare different products in order to eventually make a choice between them. Therefore, models able to predict the user rating from the text review are critically important.

  Chevalier and Mayzlin also analyse the distribution of ratings in online reviews and come to the same conclusion: the resulting data presents an asymmetric bimodal distribution, where reviews are overwhelmingly positive.

  Pang, Lee and Vaithyanathan approach this predictive task as an opinion mining problem enabling to automatically distinguish between positive and negative reviews. In order to determine the reviews polarity, the authors use text classification techniques by training and testing binary classifiers on movie reviews containing 36.6% of negative reviews and 63.4% of positive reviews. On the top of that, they also try to identify appropriate features to enhance the performance of the classifiers.

  Dave, Lawrence, and Pennock also deal with the issue of class imbalance with a majority of positive reviews and show similar results. SVM outperforms Naïve Bayes with an accuracy greater than 85% and the implementation of part-of-speech as well as stemming is also ineffective.

Nevertheless, while the previous research led to better results with unigrams, this study shows that bigrams turn out to be more effective at capturing context than unigrams in the specific case of their datasets.

In a slightly different perspective, McAuley and Leskovec intend to understand hidden dimensions of customer's opinions to better predict the numerical ratings. They also find out that the user information is a rich source of information to take into account when predicting ratings with review text.

On the other hand, presented a model which predicts the star rating of a review using sentiment dictionaries. Like most polarity determining approaches, the paper uses the unigram model to represent text. The unigram model often fails to capture phrase patterns properly which leads to polarity incoherence. To overcome this drawback, the authors also employ a n-gram model. But this way of representing text vectors leads to the creation of large sparse matrices that are space inefficient known as n-gram sparsity bottlenecks.

To overcome the problem of polarity incoherence, introduces the Bag of opinions model. This model overcomes the limitations of the unigram and n-gram models. It has 3 components: root word, modifiers and negation words. Each opinion from the corpora of reviews is assigned a score using ridge regression method. At the end, a final score is calculated by combining all the independent scores of all the opinions.

Memory-based and model-based are the two most popular methods for collaborative filtering. However, both systems take a sparse user-product rating matrix as an input and may fail to capture the quality of the users and their reviews.

In a similar project 'Using Properties of the Amazon Graph to Better Understand Reviews' proposed by Leon, Vasant, Sheldon, the predicted feedback score was represented with a linear combination of the following features: Original Rating, Review Helpfulness Count, Review Unhelpfulness Count, Review Helpfulness Ratio, Reviewer Helpfulness Ratio, Reviewer Bias, Reviewer Expertise, Reviewer Clustering, Reviewer Betweenness, and Reviewer Degree. However, the goal of their project is to predict the rating of a customer who reviews the same product more than once, while our project is to predict the user's first-time rating score.

Also, they use stochastic gradient descent to minimize the mean square error to capture the weights of such features.

- ## Motivation for the Problem Undertaken

  The project was the first provided to me by Flip Robo Technologies as a part of the internship programme. The exposure to real world data and the opportunity to deploy my skillset in solving a real time problem has been the primary motivation.

  Data needed for this project is require to scrap from E-commerce platform and data cleaning operation over it. Features derived from textual reviews are used to predict its corresponding star ratings. To accomplish it, the prediction problem is transformed into a multi-class classification task to classify reviews to one of the five classes corresponding to its star rating. Getting an overall sense of a textual review could in turn improve consumer experience. However, the motivation for taking this project was that it is relatively a new field of research.

Analytical Problem Framing

- Mathematical/ Analytical Modelling of the Problem

  In order to apply text classification, the unstructured format of text has to be converted into a structured format for the simple reason that it is much easier for computer to deal with numbers than text. This is mainly achieved by projecting the textual contents into Vector Space Model, where text data is converted into vectors of numbers.

  In the field of text classification, documents are commonly treated like a Bag-of-Words, meaning that each word is independent from the others that are present in the document. They are examined without regard to grammar neither to the word order. In such a model, the term frequency (occurrence of each word) is used as a feature in order to train the classifier. However, using the term frequency implies that all terms are considered equally important. As its name suggests, the term frequency simply weights each term based on their occurrence frequency and does not take the discriminatory power of terms into account. To address this problem and penalize words that are too frequent, each word is given a term frequency inverse document frequency (tf-idf) score which is defined as follow:

$$tf - idf_{t,d} = tf_{t,d} * idf_t$$

where:

- $tf_{t,d} = \frac{n_{t,d}}{\sum_k n_{k,d}}$ with $n_{t,d}$ the number of term t contained in a document d, and $\sum_k n_{k,d}$ the total number of terms k in the document d
- $idf_t = \log \frac{N}{df_t}$ with N the total number of documents and df_t the number of documents containing the term t

- Data Sources and their formats

  Data is collected from Amazon.in using selenium and saved in CSV file. Around 20000 Reviews are collected for this project.

```
df = pd.read_csv('Smartphone.csv')
df
```

	Unnamed: 0	Product_Review	Ratings
0	0	Camera : Great. No words for it's outstanding ...	4.0
1	1	This is a very good budget phone from Samsung ...	4.0
2	2	Screen (5/5)Camera (3.8/5)Battery (4.5/5)RAM m...	4.0
3	3	I have always liked black, this time by mistak...	4.0
4	4	IF YOU TALK ABOUT LIKE ... THEN I SAY ... BATT...	4.0
...	...	...	...
19995	19995	Been using the phone for a while now and it's ...	5.0
19996	19996	Camera should be improved. Otherwise phone is ...	4.0
19997	19997	But RAM plus not available in 4gb varient	4.0
19998	19998	Thank you Amazon ☺Product is good 👍Dilivery B...	4.0
19999	19999	Bought it for 10k.Maybe my best purchase after...	4.0

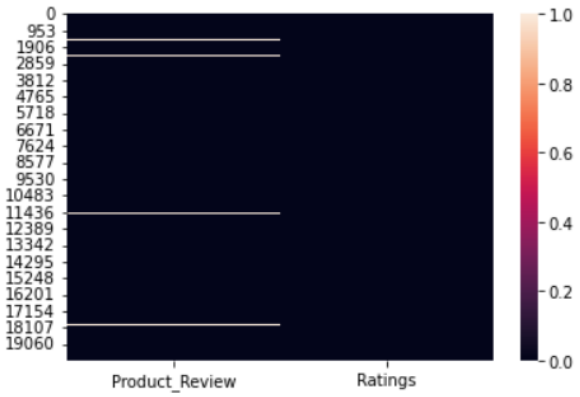
20000 rows × 3 columns

This is multi-classification problem and Rating is our target feature class to be predicated in this project. There are five different categories in feature target i.e., The rating is out 5 stars and it only has 5 options available 1 star, 2 stars, 3 stars, 4 stars, 5 stars.

```
#Checking for missing data
df.isnull().sum()
```

```
Product_Review    550
Ratings             0
dtype: int64
```

```
sns.heatmap(df.isnull());
```



There are some missing values in product review. The datatype of Product review is object while datatypes of Ratings is float.

- Data Pre-processing Done

The dataset is large and it may contain some data error. In order to reach clean, error free data some data cleaning & data pre-processing performed data.

- Missing Value Imputation: Missing value in product reviews are replace with 'Review Not Available'.

```
df['Product_Review'].fillna('Review Not Available',inplace=True)
```

```
df.isnull().sum()
```

```
Product_Review    0
Ratings           0
dtype: int64
```

- Data is pre-processed using the following techniques:

  1. Convert the text to lowercase
  2. Remove the punctuations, digits and special characters
  3. Tokenize the text, filter out the adjectives used in the review and create a new column in data frame
  4. Remove the stop words
  5. Stemming and Lemmatising
  6. Applying Text Vectorization to convert text into numeric

• Data Inputs- Logic- Output Relationships

The dataset consists of 2 features with a label. The features are independent and label is dependent as our label varies the values (text) of our independent variable's changes. Using word cloud, we can see most occurring word for different categories.

• Hardware and Software Requirements and Tools Used

Hardware Used -

1. Processor — Intel i5 processor with 2.4GHZ
2. RAM — 4 GB
3. GPU — 2GB AMD Radeon Graphics card

Software utilised -

1.Anaconda – Jupyter Notebook

2. Selenium – Web scraping

Libraries Used – General library for data wrangling & visualisation

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

Libraries used for Text Mining / Text Analytics are:

```
import re
import string
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import SnowballStemmer, WordNetLemmatizer
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from wordcloud import WordCloud
```

Libraries used for web scraping data from e-commerce website are:

```
import selenium
import requests
import bs4
import time
import pandas as pd
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.action_chains import ActionChains
from selenium.webdriver.common.keys import Keys
import warnings
warnings.filterwarnings('ignore')
from selenium.common.exceptions import StaleElementReferenceException, NoSuchElementException
from matplotlib import pyplot as plt
import matplotlib.image as mpimg
import numpy as np
import urllib
import os
```

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

  First part of problem solving is to scrap data from amazon.in website which we already done. Second is performing text mining operation to convert textual review in ML algorithm useable form. Third part of problem building machine learning model to predict rating on review. This problem can be solve using classification-based machine learning algorithm like logistics regression. Further Hyperparameter tuning performed to build more accurate model out of best model.

- Testing of Identified Approaches (Algorithms)

  The different classification algorithm used in this project to build ML model are as below:

  → Random Forest classifier
  → Decision Tree classifier
  → Logistics Regression
  → AdaBoost Classifier
  → Gradient Boosting Classifier
  → KNeighbors Classifier
  → Support Vector Machine Classifier
  → ExtraTreesClassifier

- Run and Evaluate selected models

Logistic Regression

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import r2_score,confusion_matrix,classification_report,mean_absolute_error,mean_squared_error

LOR = LogisticRegression()
LOR.fit(X_train,Y_train)

# Prediction
predLOR = LOR.predict(X_test)
print('R2 Score :',r2_score(Y_test,predLOR))

# Mean Absolute Error(MAE)
print('Mean Absolute Error(MAE)',mean_absolute_error(Y_test,predLOR))

# Mean Squared Error(MSE)
print('Mean Squared Error',mean_squared_error(Y_test,predLOR))

# Root Mean Squared Error (RMSE)
print('Root Mean Squared Error',np.sqrt(mean_squared_error(Y_test,predLOR)))

print("-----------------------------------------------------")
# Accuracy Score
print(accuracy_score(Y_test, predLOR))
print("-----------------------------------------------------")
# Confusion Matrix
print(confusion_matrix(Y_test, predLOR))
print("-----------------------------------------------------")
# Classification Report
print(classification_report(Y_test,predLOR))
```

```
R2 Score : 0.6979908443540183
Mean Absolute Error(MAE) 0.021166666666666667
Mean Squared Error 0.022166666666666668
Root Mean Squared Error 0.14888474289418197
-----------------------------------------------------
0.9793333333333333
-----------------------------------------------------
[[  58   36    0]
 [   2 5534   12]
 [   3   71  284]]
-----------------------------------------------------
              precision    recall  f1-score   support

         3.0       0.92      0.62      0.74        94
         4.0       0.98      1.00      0.99      5548
         5.0       0.96      0.79      0.87       358

    accuracy                           0.98      6000
   macro avg       0.95      0.80      0.87      6000
weighted avg       0.98      0.98      0.98      6000
```

```
from sklearn.model_selection import cross_val_score
CVscore = cross_val_score(LOR, X, Y, cv =5)
print('\033[1m'+'Cross Validation Score', LOR, ':'+'\033[0m\n')
print("CVScore :" ,CVscore)
print("Mean CV Score :",CVscore.mean())
print("Std deviation :",CVscore.std())
```

Cross Validation Score LogisticRegression() :

```
CVScore : [0.89375 0.98375 0.98425 0.92875 0.9325 ]
Mean CV Score : 0.9446
Std deviation : 0.03489469873777389
```

RandomForest Classifier

```
from sklearn.ensemble import RandomForestClassifier

# Checking accuracy for Random Forest Classifier
RFC = RandomForestClassifier()
RFC.fit(X_train,Y_train)

# [Prediction]
predRFC = RFC.predict(X_test)
print('R2 Score:',r2_score(Y_test,predRFC))

# Mean Absolute Error(MAE)
print('Mean Absolute Error',mean_absolute_error(Y_test,predRFC))

# Mean Squared Error(MSE)
print('Mean Squared Error',mean_squared_error(Y_test,predRFC))

# Root Mean Squared Error (RMSE)
print('Root Mean Squared Error',np.sqrt(mean_squared_error(Y_test,predRFC)))
print("-----------------------------------------------------")
# Accuracy Score
print('Accuracy Score: ',accuracy_score(Y_test, predRFC))
print("-----------------------------------------------------")
# Confusion Matrix
print('Confusion Matrix:\n',confusion_matrix(Y_test, predRFC))
print("-----------------------------------------------------")
# Classification Report
print(classification_report(Y_test,predRFC))
```

```
R2 Score: 0.6979908443540183
Mean Absolute Error 0.021166666666666667
Mean Squared Error 0.022166666666666668
Root Mean Squared Error 0.14888474289418197
---------------------------------------------------
Accuracy Score:  0.9793333333333333
---------------------------------------------------
Confusion Matrix:
 [[  58   36    0]
 [   2 5534   12]
 [   3   71  284]]
---------------------------------------------------
              precision    recall  f1-score   support

         3.0       0.92      0.62      0.74        94
         4.0       0.98      1.00      0.99      5548
         5.0       0.96      0.79      0.87       358

    accuracy                           0.98      6000
   macro avg       0.95      0.80      0.87      6000
weighted avg       0.98      0.98      0.98      6000
```

```
from sklearn.model_selection import cross_val_score
CVscore = cross_val_score(RFC, X, Y, cv =5)
print('\033[1m'+'Cross Validation Score', RFC, ':'+'\033[0m\n')
print("CVScore :" ,CVscore)
print("Mean CV Score :",CVscore.mean())
print("Std deviation :",CVscore.std())
```

```
Cross Validation Score RandomForestClassifier() :

CVScore : [0.895   0.98375 0.98425 0.942   0.9325 ]
Mean CV Score : 0.9475
Std deviation : 0.03369310018386553
```

DecisionTree Classifier

```
from sklearn.tree import DecisionTreeClassifier

# Checking accuracy for Decision Tree Classifier
DTC = DecisionTreeClassifier()
DTC.fit(X_train,Y_train)

# [Prediction]
predDTC = DTC.predict(X_test)
print('R2 Score:',r2_score(Y_test,predDTC))

# Mean Absolute Error(MAE)
print('Mean Absolute Error',mean_absolute_error(Y_test,predDTC))

# Mean Squared Error(MSE)
print('Mean Squared Error',mean_squared_error(Y_test,predDTC))

# Root Mean Squared Error (RMSE)
print('Root Mean Squared Error',np.sqrt(mean_squared_error(Y_test,predDTC)))
print("---------------------------------------------------")
# Accuracy Score
print('Accuracy Score: ',accuracy_score(Y_test, predDTC))
print("---------------------------------------------------")
# Confusion Matrix
print('Confusion Matrix:\n',confusion_matrix(Y_test, predDTC))
print("---------------------------------------------------")
# Classification Report
print(classification_report(Y_test,predDTC))
```

```
R2 Score: 0.6979908443540183
Mean Absolute Error 0.021166666666666667
Mean Squared Error 0.022166666666666668
Root Mean Squared Error 0.14888474289418197
----------------------------------------------------
Accuracy Score:  0.9793333333333333
----------------------------------------------------
Confusion Matrix:
 [[  58   36    0]
 [   2 5534   12]
 [   3   71  284]]
----------------------------------------------------
              precision    recall  f1-score   support

         3.0       0.92      0.62      0.74        94
         4.0       0.98      1.00      0.99      5548
         5.0       0.96      0.79      0.87       358

    accuracy                           0.98      6000
   macro avg       0.95      0.80      0.87      6000
weighted avg       0.98      0.98      0.98      6000
```

```python
from sklearn.model_selection import cross_val_score
CVscore = cross_val_score(DTC, X, Y, cv =5)
print('\033[1m'+'Cross Validation Score', DTC, ':'+'\033[0m\n')
print("CVScore :" ,CVscore)
print("Mean CV Score :",CVscore.mean())
print("Std deviation :",CVscore.std())
```

Cross Validation Score DecisionTreeClassifier() :

```
CVScore : [0.845   0.97125 0.973   0.867   0.92   ]
Mean CV Score : 0.91525
Std deviation : 0.05245331257413586
```

KNeighborsClassifier

```python
from sklearn.neighbors import KNeighborsClassifier

KNN = KNeighborsClassifier()
KNN.fit(X_train,Y_train)

# [Prediction]
predKNN = KNN.predict(X_test)
print('R2 Score:',r2_score(Y_test,predKNN))

# Mean Absolute Error(MAE)
print('Mean Absolute Error',mean_absolute_error(Y_test,predKNN))

# Mean Squared Error(MSE)
print('Mean Squared Error',mean_squared_error(Y_test,predKNN))

# Root Mean Squared Error (RMSE)
print('Root Mean Squared Error',np.sqrt(mean_squared_error(Y_test,predKNN)))
print("----------------------------------------------------")
# Accuracy Score
print('Accuracy Score: ',accuracy_score(Y_test, predKNN))
print("----------------------------------------------------")
# Confusion Matrix
print('Confusion Matrix:\n',confusion_matrix(Y_test, predKNN))
print("----------------------------------------------------")
# Classification Report
print(classification_report(Y_test,predKNN))
```

```
R2 Score: 0.6979908443540183
Mean Absolute Error 0.021166666666666667
Mean Squared Error 0.022166666666666668
Root Mean Squared Error 0.14888474289418197
---------------------------------------------------------
Accuracy Score:  0.9793333333333333
---------------------------------------------------------
Confusion Matrix:
 [[  58   36    0]
 [   2 5534   12]
 [   3   71  284]]
---------------------------------------------------------
              precision    recall  f1-score   support

         3.0       0.92      0.62      0.74        94
         4.0       0.98      1.00      0.99      5548
         5.0       0.96      0.79      0.87       358

    accuracy                           0.98      6000
   macro avg       0.95      0.80      0.87      6000
weighted avg       0.98      0.98      0.98      6000
```

```
from sklearn.model_selection import cross_val_score
CVscore = cross_val_score(KNN, X, Y, cv =5)
print('\033[1m'+'Cross Validation Score', KNN, ':'+'\033[0m\n')
print("CVScore :" ,CVscore)
print("Mean CV Score :",CVscore.mean())
print("Std deviation :",CVscore.std())
```

Cross Validation Score KNeighborsClassifier() :

```
CVScore : [0.895   0.98375 0.94525 0.9175  0.9325 ]
Mean CV Score : 0.9348000000000001
Std deviation : 0.029644308053992426
```

AdaBoost Classifier

```
from sklearn.ensemble import AdaBoostClassifier
ADA = AdaBoostClassifier()
ADA.fit(X_train,Y_train)

# [Prediction]
predADA = ADA.predict(X_test)
print('R2 Score:',r2_score(Y_test,predADA))

# Mean Absolute Error(MAE)
print('Mean Absolute Error',mean_absolute_error(Y_test,predADA))

# Mean Squared Error(MSE)
print('Mean Squared Error',mean_squared_error(Y_test,predADA))

# Root Mean Squared Error (RMSE)
print('Root Mean Squared Error',np.sqrt(mean_squared_error(Y_test,predADA)))
print("---------------------------------------------------------")
# Accuracy Score
print('Accuracy Score: ',accuracy_score(Y_test, predADA))
print("---------------------------------------------------------")
# Confusion Matrix
print('Confusion Matrix:\n',confusion_matrix(Y_test, predADA))
print("---------------------------------------------------------")
# Classification Report
print(classification_report(Y_test,predADA))
```

```
R2 Score: 0.028120912658043618
Mean Absolute Error 0.07033333333333333
Mean Squared Error 0.07133333333333333
Root Mean Squared Error 0.26708300832013504
----------------------------------------------------
Accuracy Score:  0.9301666666666667
----------------------------------------------------
Confusion Matrix:
 [[  52   42    0]
 [  53 5460   35]
 [   3  286   69]]
----------------------------------------------------
              precision    recall  f1-score   support

         3.0       0.48      0.55      0.51        94
         4.0       0.94      0.98      0.96      5548
         5.0       0.66      0.19      0.30       358

    accuracy                           0.93      6000
   macro avg       0.70      0.58      0.59      6000
weighted avg       0.92      0.93      0.92      6000
```

```
from sklearn.model_selection import cross_val_score
CVscore = cross_val_score(ADA, X, Y, cv =5)
print('\033[1m'+'Cross Validation Score', ADA, ':'+'\033[0m\n')
print("CVScore :" ,CVscore)
print("Mean CV Score :",CVscore.mean())
print("Std deviation :",CVscore.std())
```

Cross Validation Score AdaBoostClassifier() :

```
CVScore : [0.84125 0.917   0.9515  0.9325  0.9325 ]
Mean CV Score : 0.9149499999999999
Std deviation : 0.03843904265197039
```

GradientBoostingClassifier

```
from sklearn.ensemble import GradientBoostingClassifier
GBC = GradientBoostingClassifier()
GBC.fit(X_train,Y_train)

# [Prediction]
predGBC = GBC.predict(X_test)
print('R2 Score:',r2_score(Y_test,predGBC))

# Mean Absolute Error(MAE)
print('Mean Absolute Error',mean_absolute_error(Y_test,predGBC))

# Mean Squared Error(MSE)
print('Mean Squared Error',mean_squared_error(Y_test,predGBC))

# Root Mean Squared Error (RMSE)
print('Root Mean Squared Error',np.sqrt(mean_squared_error(Y_test,predGBC)))
print("----------------------------------------------------")
# Accuracy Score
print('Accuracy Score: ',accuracy_score(Y_test, predGBC))
print("----------------------------------------------------")
# Confusion Matrix
print('Confusion Matrix:\n',confusion_matrix(Y_test, predGBC))
print("----------------------------------------------------")
# Classification Report
print(classification_report(Y_test,predGBC))
```

```
R2 Score: 0.6979908443540183
Mean Absolute Error 0.021166666666666667
Mean Squared Error 0.022166666666666668
Root Mean Squared Error 0.14888474289418197
----------------------------------------------------
Accuracy Score:  0.9793333333333333
----------------------------------------------------
Confusion Matrix:
 [[  58   36    0]
 [   2 5534   12]
 [   3   71  284]]
----------------------------------------------------
              precision    recall  f1-score   support

         3.0       0.92      0.62      0.74        94
         4.0       0.98      1.00      0.99      5548
         5.0       0.96      0.79      0.87       358

    accuracy                           0.98      6000
   macro avg       0.95      0.80      0.87      6000
weighted avg       0.98      0.98      0.98      6000
```

```
from sklearn.model_selection import cross_val_score
CVscore = cross_val_score(GBC, X, Y, cv =5)
print('\033[1m'+'Cross Validation Score', GBC, ':'+'\033[0m\n')
print("CVScore :" ,CVscore)
print("Mean CV Score :",CVscore.mean())
print("Std deviation :",CVscore.std())
```

```
Cross Validation Score GradientBoostingClassifier() :

CVScore : [0.895   0.98375 0.973   0.9045  0.9325 ]
Mean CV Score : 0.93775
Std deviation : 0.035550668066859165
```

Support Vector Machine

```
from sklearn.svm import SVC

# Checking accuracy for Support Vector Machine Classifier
svc = SVC()
svc.fit(X_train,Y_train)

# [Prediction]
predsvc = svc.predict(X_test)
print('R2 Score:',r2_score(Y_test,predsvc))

# Mean Absolute Error(MAE)
print('Mean Absolute Error',mean_absolute_error(Y_test,predsvc))

# Mean Squared Error(MSE)
print('Mean Squared Error',mean_squared_error(Y_test,predsvc))

# Root Mean Squared Error (RMSE)
print('Root Mean Squared Error',np.sqrt(mean_squared_error(Y_test,predsvc)))
print("----------------------------------------------------")
# Accuracy Score
print('Accuracy Score: ',accuracy_score(Y_test, predsvc))
print("----------------------------------------------------")
# Confusion Matrix
print('Confusion Matrix:\n',confusion_matrix(Y_test, predsvc))
print("----------------------------------------------------")
# Classification Report
print(classification_report(Y_test,predsvc))
```

```
R2 Score: 0.6979908443540183
Mean Absolute Error 0.021166666666666667
Mean Squared Error 0.022166666666666668
Root Mean Squared Error 0.14888474289418197
---------------------------------------------------
Accuracy Score:  0.9793333333333333
---------------------------------------------------
Confusion Matrix:
 [[  58   36    0]
 [   2 5534   12]
 [   3   71  284]]
---------------------------------------------------
              precision    recall  f1-score   support

         3.0       0.92      0.62      0.74        94
         4.0       0.98      1.00      0.99      5548
         5.0       0.96      0.79      0.87       358

    accuracy                           0.98      6000
   macro avg       0.95      0.80      0.87      6000
weighted avg       0.98      0.98      0.98      6000
```

```
from sklearn.model_selection import cross_val_score
CVscore = cross_val_score(svc, X, Y, cv =5)
print('\033[1m'+'Cross Validation Score', svc, ':'+'\033[0m\n')
print("CVScore :" ,CVscore)
print("Mean CV Score :",CVscore.mean())
print("Std deviation :",CVscore.std())
```

```
Cross Validation Score SVC() :

CVScore : [0.895   0.98375 0.98425 0.942   0.9325 ]
Mean CV Score : 0.9475
Std deviation : 0.03369310018386553
```

ExtraTreesClassifier

```
from sklearn.ensemble import ExtraTreesClassifier

# Checking accuracy for ExtraTreesClassifier
ETC = ExtraTreesClassifier()
ETC.fit(X_train,Y_train)

# [Prediction]
predETC = ETC.predict(X_test)
print('R2 Score:',r2_score(Y_test,predETC))

# Mean Absolute Error(MAE)
print('Mean Absolute Error',mean_absolute_error(Y_test,predETC))

# Mean Squared Error(MSE)
print('Mean Squared Error',mean_squared_error(Y_test,predETC))

# Root Mean Squared Error (RMSE)
print('Root Mean Squared Error',np.sqrt(mean_squared_error(Y_test,predETC)))
print("---------------------------------------------------")
# Accuracy Score
print('Accuracy Score: ',accuracy_score(Y_test, predETC))
print("---------------------------------------------------")
# Confusion Matrix
print('Confusion Matrix:\n',confusion_matrix(Y_test, predETC))
print("---------------------------------------------------")
# Classification Report
print(classification_report(Y_test,predETC))
```

```
R2 Score: 0.6979908443540183
Mean Absolute Error 0.021166666666666667
Mean Squared Error 0.022166666666666668
Root Mean Squared Error 0.14888474289418197
-----------------------------------------------------
Accuracy Score:  0.9793333333333333
-----------------------------------------------------
Confusion Matrix:
 [[  58   36    0]
 [   2 5534   12]
 [   3   71  284]]
-----------------------------------------------------
              precision    recall  f1-score   support

         3.0       0.92      0.62      0.74        94
         4.0       0.98      1.00      0.99      5548
         5.0       0.96      0.79      0.87       358

    accuracy                           0.98      6000
   macro avg       0.95      0.80      0.87      6000
weighted avg       0.98      0.98      0.98      6000
```

```
from sklearn.model_selection import cross_val_score
CVscore = cross_val_score(ETC, X, Y, cv =5)
print('\033[1m'+'Cross Validation Score', ETC, ':'+'\033[0m\n')
print("CVScore :" ,CVscore)
print("Mean CV Score :",CVscore.mean())
print("Std deviation :",CVscore.std())
```

```
Cross Validation Score ExtraTreesClassifier() :

CVScore : [0.8825  0.98375 0.98425 0.942   0.9325 ]
Mean CV Score : 0.945
Std deviation : 0.037719027028808694
```

3-fold Cross validation performed over all model. We can see that Random Forest Classifier gives us good Accuracy and maximum f1 score along with best Cross-validation score. Hyperparameter tuning is applied over Random Forest model and used it as final model.

Cross Validation Score

```
from sklearn.model_selection import cross_val_score

#cv score for Logistic Regression
print('Logistic Regression',cross_val_score(LOR,X,Y,cv=3).mean())

# cv score for Decision Tree Classifier
print('Decision Tree Classifier',cross_val_score(DTC,X,Y,cv=3).mean())

# cv score for KNeighbors Classifier
print('KNeighbors Classifier',cross_val_score(KNN,X,Y,cv=3).mean())

# cv score for Extra Trees Classifier
print('Extra Trees Classifier',cross_val_score(ETC,X,Y,cv=3).mean())

# cv score for AdaBoosting Classifier
print('AdaBoosting Classifier:',cross_val_score(ADA,X,Y,cv=3).mean())

# cv score for Random Forest Classifier
print('Random Forest Classifier',cross_val_score(RFC,X,Y,cv=3).mean())

# cv score for Gradient Boosting Classifier
print('Gradient Boosting Classifier',cross_val_score(GBC,X,Y,cv=3).mean())

# cv score for Support Vector  Classifier
print('Support Vector  Classifier',cross_val_score(svc,X,Y,cv=3).mean())
```

```
Logistic Regression 0.9300002746637346
Decision Tree Classifier 0.9035008495599822
KNeighbors Classifier 0.9231006196464854
Extra Trees Classifier 0.9297502871631096
AdaBoosting Classifier: 0.8778022395009862
Random Forest Classifier 0.934250062174359
Gradient Boosting Classifier 0.9252505121518602
Support Vector  Classifier 0.9297502871631096
```

Hyperparameter Tuning

```
parameters = {'criterion' : ['gini','entropy'],
              'random_state' : [10, 50, 1000],
              'max_depth' : [0, 10, 20],
              'n_jobs' : [-2, -1, 1],
              'n_estimators' : [50,100, 200, 300]}
```

```
from sklearn.model_selection import GridSearchCV
GCV=GridSearchCV(RandomForestClassifier(),parameters,cv=3)
GCV.fit(X_train,Y_train)
```

```
GridSearchCV(cv=3, estimator=RandomForestClassifier(),
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [0, 10, 20],
                         'n_estimators': [50, 100, 200, 300],
                         'n_jobs': [-2, -1, 1],
                         'random_state': [10, 50, 1000]})
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
GCV.best_params_
```

```
{'criterion': 'gini',
 'max_depth': 20,
 'n_estimators': 50,
 'n_jobs': 1,
 'random_state': 50}
```

```
Rating = RandomForestClassifier(criterion='gini', max_depth=20, n_estimators=50, n_jobs=1, random_state=50)
Rating.fit(X_train, Y_train)
pred = Rating.predict(X_test)
print('\033[1m'+'Final Random Forest Classifier Model'+'\033[0m')
print('\033[1m'+'Accuracy Score :'+'\033[0m\n', accuracy_score(Y_test, pred))
print('\n')
print('\033[1m'+'Confusion matrix of Random Forest Classifier :'+'\033[0m \n',confusion_matrix(Y_test, pred))
print('\n')
print('\033[1m'+'Classification Report of Random Forest Classifier'+'\033[0m \n',classification_report(Y_test, pred))
```

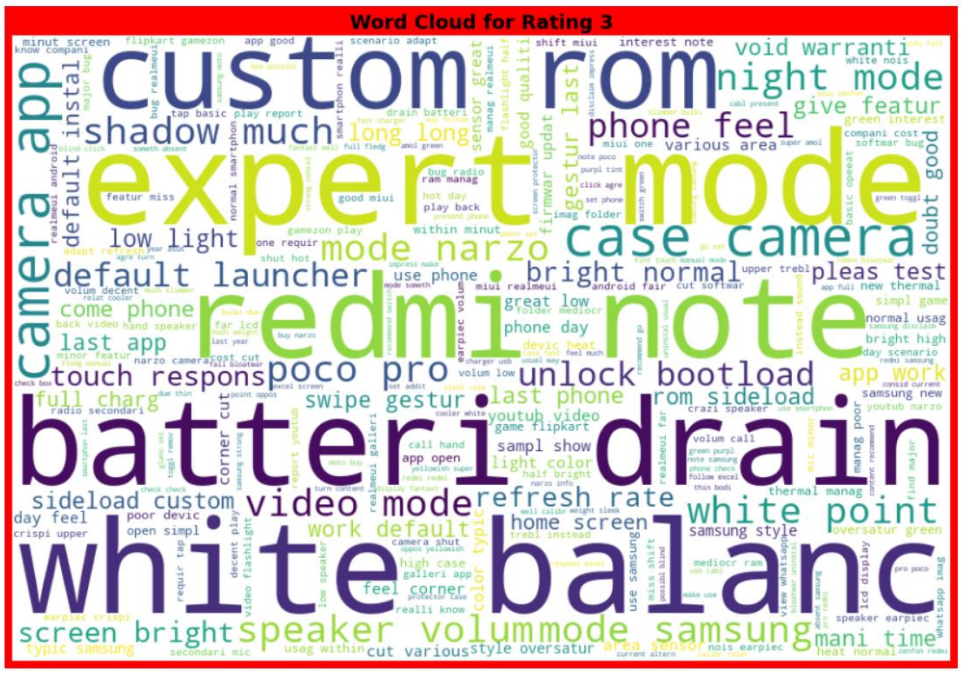
- Key Metrics for success in solving problem under consideration
    a. Precision can be seen as a measure of quality; higher precision means that an algorithm returns more relevant results than irrelevant ones.
    b. Recall is used as a measure of quantity and high recall means that an algorithm returns most of the relevant results.
    c. Accuracy score is used when the True Positives and True negatives are more important. Accuracy can be used when the class distribution is similar.
    d. F1-score is used when the False Negatives and False Positives are crucial. While F1-score is a better metric when there are imbalanced classes.
    e. Cross validation Score: To run cross-validation on multiple metrics and also to return train scores, fit times and score times. Get predictions from each split of cross-validation for diagnostic purposes. Make a scorer from a performance metric or loss function.
    f. AUC_ROC _score: ROC curve. It is a plot of the false positive rate (x-axis) versus the true positive rate (y-axis) for a number of different candidate threshold values between 0.0 and 1.0

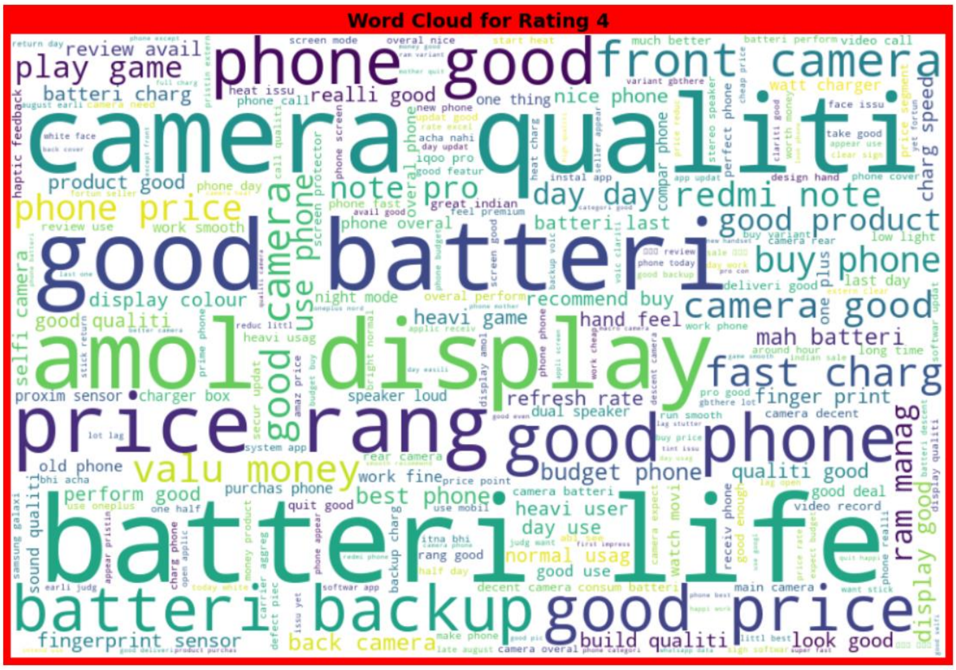
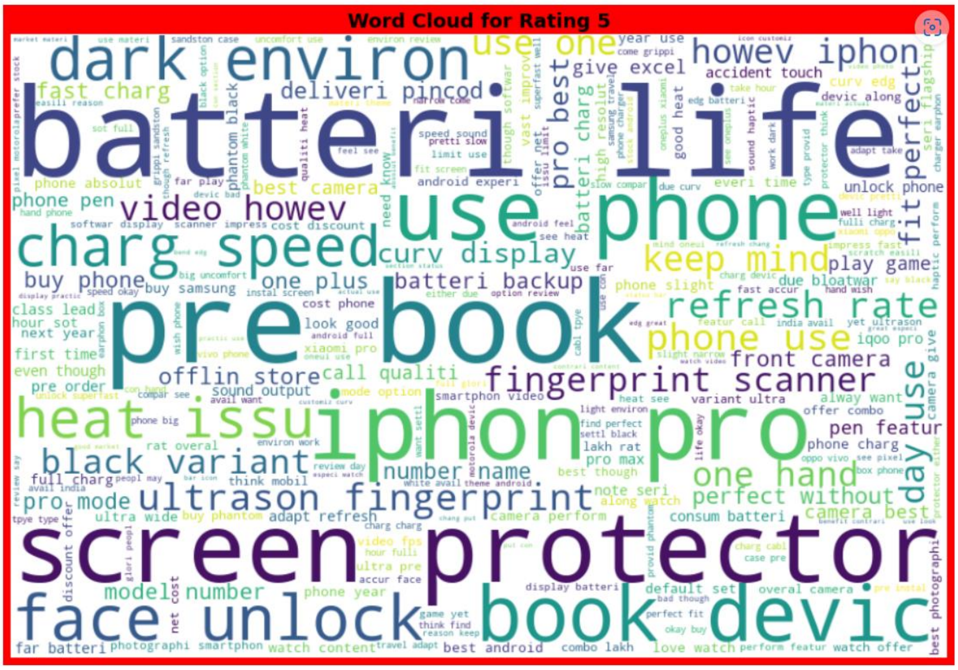
g. We have used Accuracy Score and Cross validation score as key parameter for model evaluation in this project since balancing of data is performed.

- Visualizations

Word Cloud:

a. Word Cloud is a visualization technique for text data wherein each word is picturized with its importance in the context or its frequency.
b. The more commonly the term appears within the text being analysed, the larger the word appears in the image generated.
c. The enlarged texts are the greatest number of words used there and small texts are the smaller number of words used.




Word Cloud for Rating 4
camera qualiti
good batteri
amol display
batteri life
price rang
good phone
batteri backup
good price
phone good
front camera
play game
phone price
redmi note
good product
buy phone
fast charg
valu money
camera good
day day
note pro
ram manag
display colour
recommend buy
mah batteri
refresh rate
finger print
budget phone
qualiti good
best phone
heavi user
day use
normal usag
fingerprint sensor
back camera
build qualiti
look good

Word Cloud for Rating 5
batteri life
pre book
screen protector
use phone
iphon pro
dark environ
charg speed
face unlock
book devic
heat issu
video howev
black variant
ultrason fingerprint
refresh rate
fingerprint scanner
keep mind
phone use
howev iphon
one hand
curv display
offlin store
fast charg
deliveri pincod
front camera
call qualiti
number name
perfect without
camera best

CONCLUSION

- Key Findings and Conclusions of the Study

Cross Validation Score

```
from sklearn.model_selection import cross_val_score

#cv score for Logistic Regression
print('Logistic Regression',cross_val_score(LOR,X,Y,cv=3).mean())

# cv score for Decision Tree Classifier
print('Decision Tree Classifier',cross_val_score(DTC,X,Y,cv=3).mean())

# cv score for KNeighbors Classifier
print('KNeighbors Classifier',cross_val_score(KNN,X,Y,cv=3).mean())

# cv score for Extra Trees Classifier
print('Extra Trees Classifier',cross_val_score(ETC,X,Y,cv=3).mean())

# cv score for AdaBoosting Classifier
print('AdaBoosting Classifier:',cross_val_score(ADA,X,Y,cv=3).mean())

# cv score for Random Forest Classifier
print('Random Forest Classifier',cross_val_score(RFC,X,Y,cv=3).mean())

# cv score for Gradient Boosting Classifier
print('Gradient Boosting Classifier',cross_val_score(GBC,X,Y,cv=3).mean())

# cv score for Support Vector  Classifier
print('Support Vector  Classifier',cross_val_score(svc,X,Y,cv=3).mean())
```

```
Logistic Regression 0.9300002746637346
Decision Tree Classifier 0.9035008495599822
KNeighbors Classifier 0.9231006196464854
Extra Trees Classifier 0.9297502871631096
AdaBoosting Classifier: 0.8778022395009862
Random Forest Classifier 0.934250062174359
Gradient Boosting Classifier 0.9252505121518602
Support Vector  Classifier 0.9297502871631096
```

Final Model is giving us Accuracy score of 97.76% which is slightly improved compare to earlier Accuracy score of 93.42%

- Learning Outcomes of the Study in respect of Data Science
    a. Hands on chance to enhance my web scraping skillset.
    b. In this project we were able to learn various Natural language processing techniques like lemmatization, stemming, removal of Stop words.
    c. This project has demonstrated the importance of sampling effectively, modelling and predicting data.
- Limitations of this work and Scope for Future Work
    a. More input features can be scrap to build predication model.
    b. There is scope for application of advanced deep learning NLP tool to enhanced text mining operation which eventually help in building more accurate model with good cross validation score.